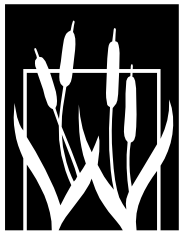

Whitemarsh
Information Systems Corporation

Data Management Program
Data Standards Architectures
and
Implementation

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: mmgorman@wiscorp.com
Web: www.wiscorp.com

Table of Contents

Tables

Acknowledgments

This material is an evolution of documents that were updated during the time frame: September 2003 through December 2004. The primary contributors were Bruce Haberkamp, James Blalock, and Michael Gorman of the Office of the CIO, United States Army. The foundational components of this work has been favorably reviewed by subject matter experts within the U.S. Department of Defense.

1.0 Introduction

This paper identifies the data standards that must be followed so that data assets have the highest degree of quality and understanding-based interoperability possible. Each data standard also contains a high level implementation process model.

Four data standards are necessary to achieve an understanding-based interoperability in an enterprise-wide net-centric database environment are:

- Authoritative Data Sources
- Information Exchange Standards Specifications
- Enterprise Identifiers
- XML

While the demand for interoperability is easy to declare, its achievement is difficult, time consuming, and laborious. The cost of not having understanding-based interoperability with minimum complexity and latency ranges from diminished information timeliness and value to fratricide.

There are actually no unsolved technical problems in achieving understanding-based interoperability. Understanding-based interoperability consists of two parts: shared value streams, and shared semantic understanding. Both of these are created from within the Communities of Interest and are expressed via the Information Exchange Standard Specifications. The role of Enterprise Identifers (EIDs) within understanding-based data interoperability is to support technology independent mechanisms to identify and locate both metadata and values (both single value and value sets). The role of Authoritative Data Sources (ADS) is to minimize the versions of the truth. Additionally ADSs enable the coordinated migration of "truth" from an originating value state through a chain of value states until the data source is either quiesced or deleted. Finally, the role of XML within this environment is to take the value streams from an originating system and to transport them to an IESS or vice versa. Embedded within the XML stream are EIDs to enable users to both understand the authority of the value sets and the supporting metadata.

Proper configuration of understanding-based interoperability requires attention to:

- Authoritative Data Sources
- Information Exchange Specification Standards
- Enterprise Identifiers
- XML data environment

If any of these four parts is missing, data will not be understanding-based interoperable. These four data standards must be based on a rock-solid, smart-engineered data management environment that consists of:

- ISO 11179 based shared data elements that can be employed as semantic templates across shared data segments within conceptual data model and in turn within logical, physical, and view data models so that differently named data that is the same can be found and vice versa.

- Shared data segments from conceptual data models that can be employed within logical data models, and, in turn, within physical and view data models so that there is consistency regarding the completeness, granularity, and precision of the collected and managed data throughout the enterprise.

- Automatic generation of names, definitions, metadata catalog entries, and XML tags so that data, regardless of its localized names and definitions can be sought out and combined as appropriate.

- Communities of interest that determine their data exchange standards (i.e., IESSs) and also the harmonization of these IESSs within domains and mission areas.

Before an explanation of each is provided, the following introduction is presented.

Authoritative Data Sources (ADS). An ADS provides common data structures and value sets that can be used with different databases. ADSs increase interoperability among databases, reducing the need for value domain translators. ADSs also contribute to easier and less costly maintenance of the database, and add to the testability of the database. ADSs are ideally constructed from shared data segments which, in turn are constructed from standard data elements.

Information Exchange Standards Specifications (IESS). An IESS is a narrowly scoped data model that facilitates data exchange and interoperability among systems within and between, communities of interest (COIs) both horizontally and vertically. These IESS data models are smaller, more manageable data models that would exist for an entire functional area. COIs can develop their own COI-internal data model for their communities but then use an IESS as the common basis and translation mechanism between members of the same COI, or across COIs.

IESSs are unlike individual Application Programming Interface (API) calls that are highly sensitive to any data model changes and would have to be modified to exchange the same data between several different systems. Ideally, IESSs are constructed from shared data segments which, in turn, are constructed from standard data elements, each of which has been developed through consensus based data standardization within a COI or a broader set of organizations.

Enterprise Identifiers (EID). An enterprise identifier is a guaranteed uniquely identifying value for an asset across an enterprise. The use of Enterprise IDs will ensure exact record-matching among heterogeneous databases even when the databases were designed independently.

Extensible Markup Language (XML). XML is a text-based format that is being used for structured document and data exchanges. XML is used in this way to describe and couch data in support of non-proprietary data exchanges that are independent of any DBMS and/or database application. The XML data streams can also be displayed through Internet browsers. XML tags need to be based on standardized data for maximum reuse.

2.0 Authoritative Data Sources (ADS)

An ADS provides common data structures and value sets that can be used with different databases. ADSs increase interoperability among databases, reducing the need for value domain translators. ADSs also contribute to easier and less costly maintenance of the database, and add to the testability of the database. ADSs are ideally constructed from shared data segments which, in turn are constructed from ISO 11179 shared data elements.

2..1 Rationale

Data, that is identified as an Authoritative Data Source (ADS) simply means it has been adjudicated as authoritative. There are different classes of authoritative data. Common examples are single value based sources such as the source for the restricted value set for Gender, Organization Name, Organization Identifier, Federal Stock Number, and the like. Another class of authoritative data sources is a set of data values that are collectively authoritative with respect to that data structure. For example, a person's legal name. It might contain, Salutation (e.g, Mr, Ms, or Mrs.), First Name, Middle Name, Last Name, and Name Suffix (such as Jr.). Other structures might be Person Skill Assessment that might contain: Person Identifier, Person Skill Type, Person Skill Assessment Value, and Person Skill Assessment Date.

ADSs are a special class of data resources. Some of them, such as reference tables, represent finite domains of specifically enumerated valid values. Reference tables perform a number of critical process-oriented functions that are vitally important in maintaining the validity and integrity of process transaction data and information that is passed and shared among application information systems such as:

- **Classification**: The values contained in reference tables are used to classify other kinds of data for a number of important process functions such as aggregation, summarization, business rule triggers and so on.

- **Transformation**: Reference tables are used to transform process transaction data from one "kind" of data to another "kind" of data, such as measure unit conversions or currency conversions.

- **Derivation Operators**: Reference table data are used in calculations with process transaction-generated data to derive values for other process type data elements. An example is applying a value represented by a Commodity Surcharge Code to Commodity Wholesale Price Amount to derive a Commodity Retail Price Amount.

- **State Transitions**: Reference tables are often used to track and/or control state changes in transaction data such as in tracking the development of a Commodity Price Amount as it begins as an "Estimated" Commodity Price Amount, to a "Calculated" Commodity

Price Amount, to a "Final Fixed" Commodity Price Amount, and then perhaps to a "Discount" Commodity Price Amount, and so on.

Entire database instances, or a subset of a database may be declared as authoritative. For example, a personnel database may be identified as the definitive source for all personnel data. Hence, authoritative. ADS do not have to be centralized. For example, a particular set of databases that contain the current configurations of units within the enterprise could both be decentralized and dynamic, while, at the same time, be authoritative.

The proliferation of uncoordinated and non-synchronized implementation of enterprise level ADSs in application information systems is a major deterrent to fully integrating automated systems across the enterprise. Data errors occur when systems employ different domain values for their reference tables. The difference in values that are shared among multiple systems is due to untimely or incomplete reference table value domain distribution or when table value domain updates are not synchronized across all systems.

It is not uncommon for there to be no effective process for either creating and maintaining enterprise level ADSs or for synchronizing their implementations.

To insure the correct and effective execution of enterprise functional business and operational processes, the ADSs that are shared at the enterprise level among application information systems must be synchronized among the application information systems that use them in functional business process queries and in business transactions that interface with other business processes across the enterprise.

2.2 Technical Construct

Each authoritative data source must be fully defined and its value sets determined. Data models in support of both single value and data structure based ADS exist and should be followed. Database applications need to be created that populate the ADS and make them available to those requiring them. As an aside, each available ADS is, in fact, an Information Exchange Specification Standard (IESS). The mode of transport of the ADS value set may be through XML, and associated with the XML elements are the appropriate set of EIDs that identify the ADS's source, authority, and meaning of both the value sets and the metadata supporting the ADS.

Every ADS should be supported by an EID so that when its semantics (which are cast as metadata) and the value instances are employed, the ADS source, represented by the ADS can be accessed for supporting information such as when the ADS was created, by whom was it adjudicated, the ANSI, ISO, or other standard that was followed for the value domains.

A key component of any ADS is its ability to map and/or convert from a "legacy" ADS to a more recent ADS. The mapping/conversion specification and process must all be available from

the ADS source in a form that can be employed against legacy databases to bring them current, or to support data mining of the legacy databases armed with the legacy ADS value sets.

ADS data and associated metadata is represented in an operational database table either through a single column or a collection of columns. In either case, the appropriate EID for the single column and/or the EID for the data structure must also be contained in the operational database table.

The SQL 1999 concept, Row Type can be employed to represent both types of ADS data as a named row type, e.g., Gender Row Type is definable as having two contained elements, Gender and Gender_EID. In the case of Person Skill Data, the column would be Person_Skill Row(Person_Identifier, Person_Skill_Type, Person_Skill_Assessment_Value, Person_Skill_Assessment_Date). The ADS for Person_Name would be Person_Name Row(Salutation, First_Name, Middle_Name, Last_Name, Name_Suffix). While there are other ways of implementing ADS, a critical implementation requirement is that the complete set of ADS data, that is, the columns that contain the actual values and also the EID that represents the source of the ADS is available.

The authoritative data source concept can also be extended to database centric processes. That is, the set of stored procedures, column and table constraints, before and after triggers, and the like. If this were done, coupled with formal definition and configuration management, then this would represent a good start towards the central definition (while decentrally distributing) and management of business rules.

2.3 Value to Net Centricity

Table 1 provides the value to Net Centricity from authoritative data sources.

Authoritative Data Source	
Net Centric Data Goal	**Value description**
Make data visible	Visibility is addressed because the data is easily known by all to be the definitive value set and is thus not obscured by a large collection of alternate forms that may not be correct or timely.
Make data accessible	Accessibility is addressed because the location of the definitive source of the information is known.
Institutionalize data management	Institutionalized because there is no need for any long search to find the one source. If the data is distributed then it is know to be properly managed and controlled.

Authoritative Data Source	
Net Centric Data Goal	**Value description**
Enable data to be understood	Given that the ADS data is fully defined within the source then this then provide only one set of definitions for the data.
Enable data to be trusted	Again, given that there is only set data for this class of data then trust (given that all updates to this data are driven to this authoritative data source) should be high and automatically follow. If the ADS is decentralized then there would need to be a proper effort expended on refreshing the definitive set of values.
Support data interoperability	Interoperability will increase because there will be a smaller quantity of translators required between any set of data to the translator to the next set of data. This will both decrease the IT processing time but will also free up resources to develop more and better systems.
Be responsive to user needs	User needs will be fewer in this area because there will be one source for the data that is appropriately made current.

Table 1. Value to Net Centricity from authoritative data sources.

2.4 ADS Implementation Process

Authoritative data sources requires these high level procedures.

- Identify high risk data structures
- Standardize high risk data structures
- Transform high risk data structures into authoritative data sources
- Incorporate enterprise identifiers
- Institute infrastructure for authoritative data sources

2.4.1 Identify High Risk Data Structures

Instead of attempting to restart failed data standardization efforts of the past, data administrators should focus their efforts on standardizing high risk data structures. In general terms, high risk data structures commonly correspond to domains of coded value sets exemplified by Country Code, Social Security Number, Unit Authorization Document Number, and National Stock Number. Examples of low risk data structures include most quantitative values, dates (these can

usually be transformed algorithmically), and non-key textual labels and descriptions used solely as reference documentation.

These data structures are high-risk by virtue of their use and representation present a significant risk to interoperability if they are not rigorously managed. High-risk data structures are almost always reference data and authoritative data sources.

High risk data structures also include common patterns of data that repeat across databases. For example, City-State-Zip. While the fields within the City-State-Zip data structure exist independently, they exist in finite sets of legal combinations. For example, Bowie, Maryland 20716 is a valid combination of City, State and Zip, while Shamokin, Pennsylvania 10210 is not. Thus, efforts need to be expended on determining these commonly used data structures, standardizing both their data structures and their value sets, and then setting them into reference data repositories that are authoritative, have unique instance identifiers, and represent common information exchange standard specifications. These high-risk data structures are actually data model templates and should reside in conceptual data models engineered for just this purpose.

Fixing this problem requires a reorientation of program focus and metrics used to measure success. Standardization resources should focus only on the data elements representing a significant risk to interoperability. Metrics should focus on the objective impact (e.g., measuring improvements in data sharing) rather than metrics measuring production (e.g., the number of data elements standardized).

2.4.2 Standardize High Risk ISO 11179 Based Shared Data Elements

At present, both the standardization of "data" (i.e., coded sets of data values) and "metadata" (i.e., information about the characteristics of such coded sets of data values) are of interest to the enterprise. For high risk ISO 11179 based shared data elements the strategy is as follows:

The first step is to standardize high risk ISO 11179 based shared data elements. For example, let's say the Logistics COI has a great interest in making all coded domain values for classes of the entity MATERIEL a part of the standard, so that everyone can refer to them in a consistent manner. This is equivalent to the standardization of ZIP codes, or ICAO codes for all the airports of the world. No matter where you fly, you can be sure that there is a code for that airport and that all international systems will route your bags to the proper destination.

A second step is to standardize high risk metadata supporting these high risk ISO 11179 based shared data elements. For example, the specifications for the entity PERSON should all be consistent, so that for C2 purposes it is agreed that the enterprise will know that the table for PERSON must contain attributes that capture HEIGHT, WEIGHT, HAIR COLOR, etc.

A third step is to standardize high risk units of measure. The actual form in which these inputs is specified should also be standardized. Thus weight should be expressed in KILOGRAMS (KG)

and have a minimum value of ZERO and a MAX-VALUE of 300, for example. This way anyone trying to enter –20KG into the system will be stopped by the application, and anyone who reads the data will know that it means KG or KILOGRAMS. If there is a requirement to express the unit of measure in American pounds, then a standard transformation can be applied for the user interface.

The actual process of creating ISO 11179 based shared data elements and shared data segments is covered in additional detail within the IESS section, 6.2.3.

2.4.3 Transform High Risk Data Structures into Authoritative Data Sources

Authoritative Data Sources consist of shared data segments and ISO 11179 based shared data elements. A data segment is a collection of attributes that may be employed as entire database table or a subset of the columns of a database table.

When a shared data segment is identified as an authoritative data source then the supporting infrastructure to collect and maintain the definitive set of shared data segment instances must be created and maintained.

2.4.4 Incorporate Enterprise Identifiers

In addition to just identifying the shared data segment as an authoritative data source, each instance should also be assigned an EID. This will ensure that every instance use of that shared data segment will be uniquely known.

A reference table is a look-up table of coded/literal data values (e.g., MD with the meaning, Maryland), containing generally static data instances (e.g., state code, country code, SSAN, blood type, etc.), and associated definitions. These look-up tables are the error checking mechanisms employed in thousands of mission critical systems. The accuracy and validity of this data are vital to knowledge workers and decision-makers.

EIDs are a means of tagging the reference data and uniquely identifying instances of authoritative data source. For example, if the authoritative source for the country code for the United States assigns an EID of 1234567890 with an official text symbol of "US," any other symbol(s) may be mapped to the EID (1234567890), provided it is the EID, and not the symbol, that gets used for data exchange (unbeknownst or transparent to the user)." EIDs provide the logical way to perform configuration management, broadcast updates, and interface with multi-national interoperability requirements.

EIDs are ideal for managing reference data (i.e., coded domains of data values). Systems using this approach could begin to demonstrate that data exchange and understanding-based data interoperability can be accomplished by passing EIDs instead of the reference data itself, which

would be preloaded. EIDs would enable users to customize the display of the data as they see fit without affecting interoperability.

If a data asset employed country codes, and had, for example, a county code and enterprise identifier that was Czechoslovakia, then an EID lookup would show that the specific instance of Czechoslovakia is no longer valid, and that it had been replaced by two new EIDs, one for the Czech Republic and another for Slovakia. If the set of all authoritative data sources are thoroughly interrelated especially as it relates to configuration management, the user would then be able to know whether data referred to an old EID (that is, Czechoslovakia) or the new EIDs (that is, the Czech Republic and another for Slovakia).

2.4.5 Institute the Infrastructure for Authoritative Data Sources

There are three major activities that must occur for authoritative data sources. The first is the establishment of the ADS infrastructure, the second is the capturing and maintain the ADS, and the third is the employment of the ADS value sets.

Establishing the ADS infrastructure requires projects that research and identify all the required ADSs, configuring the appropriate standard data elements and shared data segments and setting up the IT infrastructure for capturing and updating instances, advertising them, and enabling them to be employed in various information systems. Creating standard data elements and segments are accomplished within the IESS section that follows.

The second major step is the determination of the definitive source for an ADS. It is likely that in many cases the source may be both decentralized and conflicting. For example, there may be a centralized HR system that collects and stores person addresses. That person may, however, go to an organization and while providing a current address, be notified via a download of the "definitive" address that it does not match the one being provided. At that point, the decentralized capturing system should store the "new" address and at the same time notify the definitive ADS repository for addresses that there is a "candidate" update. At that point, the ADS repository system and the ADS authority would query the actual address holder for a confirmation of the address change. If the address change is correct, then not only must the ADS change be recorded, the system that originated the change of address must be notified, and the new address must then be queued up for propagation to all application information systems that employ that address.

The third major step, employment of ADS value sets, starts with initialization and then with a constant stream of distribution of updates. The update distribution process may be accomplished through a service oriented architecture whereby a standard demand transaction is executed by an employing system for updates from the ADS. Such a transaction may also identify the current set of ADS instances through the use of an EID stream that is provided to the ADS source. If any has a changed value set then the update would be sent back.

2.4.6 ADS Implementation Summary

Authoritative data sources are very important for a well-ordered data management environment. An examination of the data asset products that are built clearly show that these additional products mainly focus on the interaction between application information systems. One would be the ADS application information system that causes the creation of the data, and the other application information systems that employ ADS data. The data asset products would include the intersections metadata. The data asset products that should be build during ADS implementation are listed in Table 2.

Data Asset Products for Authoritative Data Sources		
Prod Ref	**Architecture Product**	**Architecture Product**
SV-1	Systems Interface Description	Business Information Systems, **Views, and Inter-view Mappings**
SV-3	Systems-Systems Matrix	Business Information Systems, **View Models, and Physical Data Models**
SV-6	Systems Data Exchange Matrix	Business Information Systems, **Views, and Inter-view Mappings**

Table 2. Identification of Data Asset Products for Authoritative Data Sources

The key consideration in the building of an ADS is the granularity that the ADS values are updated. If these values are updated, for example, weekly, then the business information system that employs ADS values should also launch ADS value sets weekly as well.

For example, of the Unites States Postal Service updated zip codes weekly and several different user systems of USPS zip codes only check for updates weekly, monthly, or quarterly, then there is a strong likelihood that USPS zip code ADS data across multiple IESS will not be synchronized and reporting errors might occur.

Another example would be different application information systems updating their personnel assignments at a different granularity. In such a case a person might either be assigned to multiple organizations simultaneously, or neither simultaneously. In short, synchronization is critical.

Major ADSs may created and maintained within their own COI. Smaller collections of ADS may be created within special COIs created exclusively for the creation and maintenance of ADSs. ADS harmonization is a very important issue. The logical data model of the ADS must be supported by conceptual data models and also ISO 11179 data elements including all supporting ISO 11179 metadata. These structures are very important because the conceptual data models and ISO 11179 data element structures will be the source of mappings to other IESS data models. These mappings will support the “where used” reports that enable enterprise data management evolution and maintenance.

2.5 Summary

Authoritative data sources are critical to coherent and consistent, enterprise-wide data management. The most essential component of any database is that its data be trusted. Without Authoritative Data Sources there will continue to be an increase in "versions of the truth." Without ADS, when a fact is updated there will be no single place where that update should ultimately go. Further, there will be no place from which the then current set of all critical reference data can be obtained. Every data value from critical enterprise facts will be "just your opinion." That's no way to run the enterprise. The whole concept of command and control proceeds from authority, doctrine and hierarchy. That applies to data and it's truth-value as well. Net-Centricity depends on high-quality, truth-based data and that proceeds from authoritative data sources.

Every ADS is essentially an IESS. Further, every ADS should have EIDs, and finally, the most common transport for an ADS will be XML. In short, these four go hand and glove and together make Net-Centricity practically possible.

3.0 Information Exchange Standard Specifications (IESS)

An IESS is a narrowly scoped data model that facilitates data exchange and interoperability among systems within and between, communities of interest (COIs) both horizontally and vertically. These IESS data models are smaller, more manageable data models that would exist for an entire functional area. COIs can develop their own COI-internal data model for their communities but then use an IESS as the common basis and translation mechanism between members of the same COI, or across COIs.

IESSs are unlike individual Application Programming Interface (API) calls that are highly sensitive to any data model changes and would have to be modified to exchange the same data between several different systems. Ideally, IESSs are constructed from shared data segments which, in turn, are constructed from standard data elements, each of which has been developed through consensus based data standardization within a COI or a broader set of organizations.

3.1 Rationale

When two organizations need to exchange information, they simply have an information exchange requirement. When a formal specification of that information exchange requirement is created it represents an information exchange specification. When the two organizations then place that information exchange specification under configuration management and establish that it should not change, for example, other than yearly, then the information exchange specification becomes an information exchange standard. Stylistically, this is called an information exchange standard specification (IESS).

Once an IESS is specified, it can be implemented any number of different ways. One way is through a database. IESSs that represent simple exchange transactions are really equivalent to an SQL View. The reason this is so is that the IESS in this case merely represents the sharing of data for a specific purpose.

Another form of an IESS is a message of the types and kinds that have been operational within U.S. Army applications for the past 20 years. The VID (Variable Message Format (VMF) Integrated Database (VID)) is essentially a database containing the specification set of a large collection IESSs all of the “messages” format.

IESSs exist within various communities of consensus. Two system-owners are a community, small, but a community nonetheless. A larger collection of users could agree on a collection of IESSs so that they can bring efficiency into their common exchange of information. Even larger communities of interest can exist as COIs become broader the ability to get semantic consensus becomes more difficult and time consuming.

The foundation layer of an IESS consists of an agreement within the Community of Interest regarding each and every IESS data model column. The agreement must extend to every

relationship between IESS data model tables, all the column constraints, table constraints, and if appropriate, assertions and triggers. The data model of every IESS should be supported, as appropriate, by EIDs so that any receiving organization can know the source of the IESS data.

Database columns required to support information exchanges must be managed as a key resource to support an information-driven, network-centric enterprise. The experience under the U.S. Department of Defense's 8320.1 Standardization process clearly indicates that:

- Management of an enterprise-wide complete inventory of database columns is not feasible.

- Only those database columns that cross system boundaries matter for the purpose of ensuring understanding-based data interoperability should be standardized as ISO 11179 based shared data elements.

- Only the organizations that participate in the development of those exchanges are knowledgeable enough and care enough about the process to make it successful.

Therefore, any management of data for the enterprise must be based on Communities of Interest (COI's), and must be focused first and foremost on the data for exchange. The U.S. Department of Defense's Multinational Interoperability Program (MIP) has existed as a COI from the middle 1990s in support of exchanging Command and Control (C2) data within NATO. It is a proven success because it has:

- An active and well scoped set of participants with documented information exchange requirements—a community of interest,

- A commitment strictly to conform to a specified set of database columns for exchange for any transaction that crosses system boundaries—the information exchange standards specification (IESS), and

- An agreement that each participant is free within the boundaries of its system to adopt any implementation of its choice.

This success in the area of understanding-based data interoperability clearly shows that this approach can deliver the performance needed to support understanding-based data interoperability, and can achieve data exchange success without running into the bureaucratic swamp that brought the DoD 8320.1 process to a halt in advance of its procedures being rescinded in May 2003.

The MIP's C2IEDM (Command and Control Information Exchange Data Model) is an IESS, and represents the information exchange standard specification consensus across a coalition of countries within NATO. The value of the C2IEDM is that it represents the participants consensus on C2 data exchanges by those countries. Whether any of the NATO nations have

actually adopted the C2IEDM data model to be "the" data model for any database application system from that country is both completely immaterial and irrelevant to the value of the C2IEDM. Its value is not that it has affected the internal database design of any country. Rather, that it represents the semantic, granularity and precision specification for exchanged C2 data.

Other very successful IESSs affects an unfortunate few of us every day. There is an ANSI standards group of State Departments of Motor Vehicles that has standardized how they both "read and write" data regarding traffic tickets. A police agency within a given state issues a traffic ticket. The data is recorded in their internal systems and then exported through the State Departments of Motor Vehicles traffic offense data format to the "home" state of the driver license holder. Some might argue the "value" of such a standard data exchange success, but, notwithstanding, it, as an IESS, exists.

3.2 Technical Construct

An information Exchange Specification Standard (IESS) is a data model consisting of one or more tables integrated via primary and foreign keys, supported by SQL based processes for column constraints, table constraints, and if appropriate, assertions and triggers. The IESS represents a shared data exchange standard. Every ADS is an IESS. IESSs exist because a collection of users within a community of interest have identified a collection of data that they wish to share. An IESS is more cost effective than a system to system interface in all but the most trivial of situations.

The source of the data within the IESS must be agreed to by the Community of Interest with respect to its authority, quality, and timeliness. Additionally, if there are alternate sources for an IESS, each source must agree to their role. That is, originating source, updating source, or even deleting source. These populating scenarios must be set within business cycles and calendars as discreetly occurring business events.

For example, if an IESS is a Employee Skill Data set, then it is likely that an human resources organization might be originating source. If during an employee's tenure another organization captures additional skill data then whether this newly captured skill data is an update to an existing set, or is to comprise a new instance for that employee must be predetermined.

IESSs are critical whenever there is a proprietary source for data. For example there may be an enterprise resource package (ERP) for human resources, finance, inventory, and the like. Each ERP vendor typically makes their ERP database schema proprietary. Whenever organizations want, for example, Logistics Data, one of following three scenarios must occur:

- System to system interfaces (point-to-point)
- Application program interfaces (API)
- Information Exchange Standard Specifications (IESS)

For every system to system interface (i.e., point-to-point), a specification for that interface must be created. and this approach has been well proven to be too inefficient and too expensive.

The second approach, an application program interface (API), too has been shown as problematic as there may be multiple ERP systems that have the same data and each will likely have a very different strategy for database access. ERP data model architectures range from a very typed-table approach where there are fewer than 50 database tables to a very explicit table approach where, for the same ERP domain, there could be several thousand database tables. In either situation, every interfacing user would have to thoroughly understand the underlaying ERP data model. Not a simple task. Further, if the ERP system changed then all the interfaces would need to be completely redone. Finally, if there were 20 ERPs, and an API were created for each then the overall architecture would be very fragile. Alternatively, if an enterprise created and mandated ERP functional area IESSs, then given standard functions supported by conformance tests, then there could be free and open competition for ERP solutions as there currently are for desktop, laptop, server computers, and to some extent, DBMSs.

The third approach, "bolts" an Information Exchange Standard Specifications (IESS) onto the ERP system. This alterative would require the creation of a comprehensive IESS data model that embraces the domain of the ERP package's shared data. This would enable a standardized approach to all ERP access regardless of subject area and/or ERP vendor, and it would also cause the creation of a standard subject area data model for that ERP domain.

The IESS approach should be the standard method of data exchange regardless of application type except in cases of severe bandwidth restrictions.

The exchanged data value streams of IESS data should, where practical, conform to XML. Additionally, the generation of the XML-ized data from the IESS should be supplemented by EIDs so that any receiving organization can know the complete metadata specification of the IESS data. If there is a restricted band-width, then the IESS transaction formats may not be in XML but in a different format similar to traditional messages.

Every IESS must be supported by a database application that can receive the XML data stream, determine its acceptability, and store it according to a set of predetermined rules. If there is to be an exchange of data between two IESS, then there must also be export functionality.

Surrounding the IESS should be a robust set of SQL views that are effectively a set of standard processes that retrieve data from the underlying IESS data model and provide it to the requesting system. These view specifications are a critical part of the IESS as they shield the data requesting organizations from having to know the underlying IESS data model.

The relationship between the IESS concept and the data architecture classes is as follows:

- Original data capture databases are unlikely to be IESS structures as they are most commonly contained completely within their own domain.

- Transaction Data Staging Area Databases are IESS structures by definition.

- Operational Data Store (ODS) database are unlikely to be IESS structures as they are too large, complex. However, IESS structures are likely to be created in support of data extracts from ODS databases.

- Data warehouse databases are also unlikely to be IESS structures as they are most commonly not-normalized, redundant, and mainly suited to report reporting and analysis.

- Reference data databases are almost always IESS structures.

In order to successfully build IESSs within the context of the enterprise so that the IESSs are both efficient and effective, there needs to be a supporting data model infrastructure of both ISO 11179 based shared data elements and shared data segments (via conceptual data models). First and foremost, at the core of an IESS is a data model.

IESSs should exist first as logical data models and then, in turn, as physical data models, which, in turn are accessed by information systems as SQL views. When these IESSs are further mapped to ISO 11179 based shared data elements and shared data segments then there can be IESS integration across various COIs, domains, and mission areas.

Thus, the full metadata context of every IESS are these five data model layers. Data modeling involves creating five distinct models of data: ISO 11179 shared data elements, and conceptual, logical, physical, and view data models. The conceptual, logical, and physical views may also be supported by diagrams that depict entities, attributes within the entities, and relationships among entities. Basically, data modeling serves as a link between business needs and system requirements. A data model represents the persistent data policy of the enterprise. Complementing data models are processes. Together data models and process models represent the enterprise's persistent data policy and procedures.

3.3 Value to Net Centricity

Table 3 provides the value to Net Centricity from Information Exchange Standard Specifications.

Information Exchange Standard Specifications	
Net Centric Data Goal	**Value Description**
Make data visible	The data will be more visible because it exists within a well engineered data structure designed for data exchange purposes.

Information Exchange Standard Specifications	
Net Centric Data Goal	**Value Description**
Make data accessible	The data will be accessible because it will be engineered according to shared community standards and will be updated in a coordinated manner across the participants of the community. It is important that IESSs be supported by EIDs so that all data about assets can be discovered and retrieved from their various IESS data stores.
Institutionalize data management	The data will be institutionalized because it will be built through community standards for semantics, granularity and precision.
Enable data to be understood	Given that there would be significantly reduced quantity of shared data sources then it is likely that there would be significantly reduced confusion. Also, because the IESS data would be based on ADS, then understandability would increase because there would be a broader consensus as to the metadata supporting the IESS. If IESSs contain EIDs as part of the IESS metadata then different IESSs can be more easily interrelated through metadata mapping.
Enable data to be trusted	Given that the IESS is based on community consensus, and ADS coupled with broadly understood semantics then the trustworthiness of the data is increased.
Support data interoperability	Again, all the characteristics cited above enable fewer translators and semantic mediators
Be responsive to user needs	If the IESS is squarely based on the needs of the communities of interest then there is a greater probability that the data will be

Table 3. Value to Net Centricity from Information Exchange Standard Specifications.

3.4 IESS Implementation Process

Accomplishing Information Exchange Standard Specifications requires these high level processes:

- Create ISO 11179 based shared data elements
- Create shared data segments
- Develop individual IESSs
- Publish IESSs

- Configuration manage IESSs
- Employ IESSs in individual projects

In general, the reason for these steps is to engineer collections of data structures that are founded on a fundamental set of business facts that, in turn, are represented within data model templates, that, in turn, are employed as the semantic source of data models that are configured into database schemas that, in turn, are deployed through DBMSs into the databases that, in turn, are used by the automated information systems.

In the case of legacy systems, however, IESSs may be distinct databases supported by their own IESS information systems that receive from and/or provide data to the legacy systems. In that case, the IESSs merely have to semantically map to the database structures of the legacy systems.

3.4.1 Create Shared Level Data Elements

ISO 11179 based shared data elements are created through two scenarios. The first is the mining of already existing data asset metadata, and the second ongoing modification of the pre-built set of ISO 11179 shared data element metadata. In the case of the DoD, the DDDS is a readily available source of data element (actually database table columns) metadata that should mined to discover and then build the ISO 11179 metadata levels. Specifically built would therefore be:

- Concepts
- Conceptual Value Domains
- Data Element Concepts
- Value Domains, Value Domain Values and mappings
- Data Element Classifications, and then
- [Shared] Data Elements

The second scenario occurs from the COI processes that cause the building of IESSs from existing legacy systems. This process involves importing legacy system physical data models and then building the IESS logical models. As these are built, the next logical step is to connect the IESS logical database tables to appropriate conceptual data model shared data segments. During this process, new ISO 11179 based shared data elements may be discovered and created through the use of adding more conceptual value domains, data element concepts, value domains, and the like.

During this analysis, design and discovery process the need to interact with other communities of interest will arise because there may be a common ISO 11179 based shared data element but with conflicting value domains. These conflicts need to be resolved or else the data cannot be shared.

As ISO 11179 based shared data elements are created along with the ISO 11179 super structure, if each such component contains abbreviations and definition fragments, then the creation of data element abbreviations and definitions can largely be automated.

3.4.2 Create Shared Data Segments

In the 1990s, the U. S. Department of Defense also embarked on the development of an overall Data Architecture (DDA). This result is a multi-hundred million dollar investment in the creation of defense related data models. While it is very likely that these models do not meet the exact data model needs of specific information systems, they do however potentially represent a high level of data standardization of the data that needs to be collected for that specific purpose.

The DDA effort failed for reasons similar to the reasons for the DDDS failure. Notwithstanding, these models appear to be highly valuable conceptual level data model templates that can be employed, as templates, throughout DoD information system databases. Thus, while a number of specific DoD information system databases may have different data models for the same data-concept, these data model templates may well be coalescing mechanisms to assess similarity and differences. Consequently, if "mined," these non-redundant data model patterns would likely form valuable and reusable collections of subjects, entities, and attributes within the conceptual data model area of the metadata repository.

In a manner similar to the discovery of ISO 11179 based shared data elements, shared data segments may too be discovered. Ultimately, it is highly likely that a virtually complete set of shared data segments will have been discovered and thereafter it will mainly be a mapping of existing shared data segments to newly formed IESSs. Once this point is reached, IESSs may well be generated through the identification and then automatic use and configuration of relevant shared data segments into IESS logical data models.

Given that the shared data segments are characterized by definitions at the entity level, and through inheritance, attributes are defined through mappings from ISO 11179 based shared data elements, then abbreviations and definitions attributes can be largely automated. If further, entities are associated with metadata catalogs, when the IESS is either mapped to or built from shared data segments, the generation of metadata catalog data can largely be automated.

There may be situations where physical data model table columns, or logical data model table columns, or conceptual data model entity attributes are more specialized then the ISO 11179 based shared data element from which they are mapped. In that case, then more specialized definitions and/or value domains can be incorporated at the conceptual, logical, or physical levels. In such cases, the complete definition would be stylistic combination of all four levels of definition, and the creation of this too could be largely automated.

3.4.3 Develop Individual IESSs

An IESS is related to some activity that requires the sharing of data. It represents shared data in support of some information requirement. As mentioned earlier, an Information Exchange Standards Specifications (IESS) expresses common meanings and relationships (semantics) in terms of two or more communities of interest (COI). While there may be many classes of data models, for example an ADS, reference data, or data warehouse, an IESS is a narrowly scoped data model to facilitate data exchange and interoperability between communities of interest (COIs). COIs can develop their own COI data model for their communities and use their IESS as the common basis and translation mechanism between them. Unlike individual API calls that need to be modified to exchange the same data between several different systems, using an IESS provides for one common translation mechanism that everyone builds to.

In an enterprise, data models are divided into two classes: architecture and "real data". The architecture data models determine the information required to build, manage, and exchange information used to generate architecture products. The “data” that the data asset products that exist explictly or implicitly within these views are traditional IT metadata. A metadata repository is an IESS of architectures and IT metadata.

These products are metadata as they provide views of where to put equipment on the battlefield, changes in organizational structures, the components and flow of information among units, etc. Examples of models in this grouping are the two versions of the C4ISR Core Architecture Data Model (CADM). The CADM is an IESS data model of metadata.

In contrast, "real data" data models represent the actual instances of materiel, locations, persons, facilities, and C2. The most universally acceptable C2 data model is the C2 Information Exchange Data Model (C2IEDM). The C2IEDM is a coalition data model, and the C2CDM is a joint data model. The C2IEDM is an IESS of C2 data.

A most significant value of the C2IEDM is that it is the C2 data model for data exchanges among C2s. As such, the C2IEDM is a core data model for the Multilateral Interoperability Program (MIP), Joint Battle Management Command and Control (JBMC2), Future Combat System (FCS), and other C2 involved programs.

IESSs are easier and less costly to maintain. Legacy systems can keep their old database schemas and map to the IESS selected for interoperability. Changes to an individual system won't affect the other systems that it must exchange data with. COIs that share information with each other can create their COI IESS without having to gain concurrence throughout the enterprise– thus it should be easier to develop and implement. For maximum interoperability, it is highly recommended that the various COI IESSs use the shared data segments (based on standardized data) and be able to map (ideally match) their structures to the core IESS for common components between models (e.g., the organization segments should match across all IESSs).

3.4.4 Publish IESSs

An IESS should be considered completed only after the following have been finished:

- Its ISO 11179 based shared data elements are all created or mapped from IESS logical database columns.
- Its shared data segments are all created or mapped from the IESS logical database columns.
- Its tables, columns, primary and foreign keys, and all necessary specialized definitions have been created.
- Its tables have all been associated with the appropriate metadata catalog data that have been inherited from shared data segments.
- Its column all have the appropriate XML tags that are either inherited from attributes or ISO 11179 based shared data elements, or specially created.
- All the value domains are properly referenced from ISO 11179 based shared data elements or shared data segment entity attributes, or specialized.
- Once the IESS's physical data model is created, and once all the sending and receiving information systems are mapped to the IESS.
- Once all the business events and calendars are all appropriately set out so that the IESS data sets are ensured of proper and timely valuation.

At this point, the IESS is ready to publish in the appropriate manner. Further, because the IESS's metadata is fully resident in the metadata repository, then its place within the context of all other IESSs can be fully known. That means that it will be quick and easy to know which IESSs have incorporated which ISO 11179 based shared data elements, and shared data segments. And which IESSs are being used by which information systems. And finally, given that an implementation of an IESS is itself a database information system, it will be easy to know the exact sequence of data appearances from which information systems into the IESS databases.

3.4.5 Configuration Manage IESSs

Since IESSs represent shared data exchanges across what could be hundreds and possibly thousands of systems and users, it is critical that once an IESS is set into place and then "counted-upon" as an authoritative source of data that it be changed only in a very controlled manner. Such control would have to involve:

- Gathering change requirements
- Configuring change approaches
- Assessing the impact of proposed changes
- Determining the resources required for IESS change
- Determining the resource impact for all information systems that have the IESS as an ADS
- Creating appropriate assessments, alternatives, and change plans
- Setting out schedules and accomplishing the change.

3.4.6 Employ IESSs in individual projects

An IESS is just the specification of shared data across a class of projects. There are three fundamental approaches to the implementation of an IESS. These are:

- Bolt-on IESS data portal
- Separate IESS database and supporting information system
- Modification of internal database schemas to conform precisely to the IESS

3.4.6.1 Bolt-On IESS Data Portal

The bolt-on IESS strategy causes the employing information system to create a separate database within their environment that is be the portal to the "outside" world for the shared data. Key among the benefits of this approach is that the "outside" world never has to know any of the internal designs of the employing system. This solution is ideal for ERP (enterprise resource planning packages). As to the database design of this bolt-on IESS, it would be preferred if all the IESS database table columns were exact clones of ISO 11179 based shared data elements and shared data segments.

A very key consideration is the granularity of the records of data. For example, suppose there was an human resources bolt-on IESS, in which the columns of data were the significant characteristics of the employee. The issue would be whether these are the characteristic values at the end of the year, quarter, month, or every time a change happened. Further, would there be history? These questions all then need to be both asked and answered.

This IESS paradigm is merely one of "harmonizing" the ERP's data to that of the enterprise. Thus, there is no process of data extraction, transformation, or loading that data into the IESS database that would have obtained its data from multiple organizations.

It may be that this particular IESS schema is very narrow and is restricted to just the data that the sending system wishes to exchange.

The process of transferring data around this environment, that is, through XML or some other format is immaterial to this example.

3.4.6.2 Separate IESS and Database System

The separate IESS database and supporting information system implies that there is a single database which has the database schema of the IESS and there is a separate database information system supporting that IESS. How data gets stored into the IESS can be through various means such as on-line updates, or through batch load or updating facilities that have been built into the sending systems. It may very well be that this particular IESS schema is quite large and comprehensive and thus represents the union of all IESS schemas that would exist across a domain of bolt-on IESSs.

In this example, there would have to be careful coordination across the various systems. For example, suppose one system was a customer management system and another system was an order management system. Further, that you could not have an order without the ordering organization first being a customer. Finally, suppose that the customer management and ordering systems were not able to interact with each other. The systems topology therefore, would have to be that new customers or customer updates would be captured and loaded into the customer management system. Then, these customer transactions would be sent to the Customer-Order-Management IESS. Then, when a new order arrived, the order system would query the Customer-Order-Management IESS database application system to ensure that there was a customer. If there was, then the order could be taken and stored in the order management system, it too would be sent to the Customer-Order-Management IESS database. If the customer did not exist then the order either could not be taken, or taken provisionally and then resolved later, or finally, an automatic control transfer to the customer system to capture the customer data, post it to the IESS, and then transfer control back to the order entry system.

Regardless of the scenario, the Customer-Order-Management IESS database would be consistent and would be available to others for retrieval. This scenario could extend to a Order Fulfillment system or a marketing system, in that they would be contributing first data requirements for the IESS database design and then rows of data.

The process of transferring data around this environment, that is, through XML or some other format is immaterial to this example.

3.4.6.3 Internalized IESS Data Model

This last example would likely be for new systems development. In this scenario the IESS schema would become the source schema for a new database application. Thus, in the previous example, if there was to be a single new comprehensive system, then the complete schema for the Customer-Order-Management IESS database would then become the complete schema for

this new system. In this case, however there would still be a need to post the collected data to the Customer-Order-Management IESS database so that all may have access to it.

3.4.7 IESS Implementation Summary

IESSs are the mechanism of data sharing. An IESS may be implemented in any of the ways set out in Section 3.4.6. And, if there are direct data exchanges based on shared data segments, the IESS may be completely virtual. Strategies for various database interface architectures is presented in Section 4.2. The data asset products that should be constructed for an IESS are listed in Table 4.

Data Asset Products for Information Exchange Standards Specifications		
Prod Ref	**Architecture Product**	**Architecture Product**
AV-1	Overview and Summary Information	Mission, **Database Domains**
AV-2	Integrated Dictionary	Business Terms, **Data Element Model**
OV-6a	Operational Rules Model	Business Functions, Business Events, **Physical Data Model, Views, Data Integrity Rules**, Business Information Systems, and **Database Objects.**
OV-7	Logical Data Model	**Data Element Model, Conceptual Data Model, and Logical Data Model**.
SV-4	Systems Functionality Description	Business Information Systems, **Database Objects Model, and Logical Data Model**
SV-6	Systems Data Exchange Matrix	Business Information Systems, **Views, and Inter-view Mappings**
SV –10a	Systems Rules Model	Business Information System, **Database Objects, Data Integrity Rules and Logical Data Model**.
SV-10b	Systems State Transition Description	Business Events and **Database Objects**
SV-11	Physical Schema	**Physical Data Model and View Data Model**

Table 4. Identification of Data Asset Products for an IESS

Information Exchange Standards Specifications (IESS) should be the focus of the first round of data standardization. The data standardization of IESSs will represent the majority of shared data. The logical data models of the IESSs should be examined to build the ISO 11179 data elements, and also these IESS logical data models should be examined to build the conceptual data models. As the conceptual data models and ISO 11179 data elements are built there will be an increased ability to have data standardization across the IESSs.

As an IESS is built, its logical data model table columns will be mapped to conceptual data model entity attributes. Additionally, the IESS logical data model table columns will be mapped to ISO 11179 data elements mainly through the mapped conceptual data model entity attributes, and exceptionally to ISO 11179 data elements directly.

Value domains are a very important part of data standardization. Value domains must be harmonized across all the IESSs that contain database table columns who's value set is derived from an ADS. There are three cases to consider when harmonizing value domains: no difference, subset, and superset. If there is no difference between a database table column's value domain and that of its parent attribute, then none needs be specified as it automatically inherits the one associated with the database table column's parent attribute or the attribute's parent ISO 11179 data element. In the case of a subset value domain, then the values that comprise the subset must be specified as a value domain value subset and then that subset assigned to the database table column. Finally, if the value domain is a superset, then the value domain of the parent attribute and possibly the value domain of the attribute's parent ISO 11179 data element must be adjusted.

It is critical that the IESS feeder and using systems are synchronized as to data precision and granularity. For example, if one feeder system is charged with producing "header" information which may be in the form of ADS type data, and other AIS feeder systems produce the "transaction" data then the header information must certainly be present prior to the transaction data.

3.5 Summary

An Information Exchange Standard Specification is merely a data-specified agreement between two or more organizations on how they will exchange data and what the data truly means when exchanged. There can be no understandability-based interoperability without IESSs. As time shrinks between data collection and required data use, the requirement for consensus based data standards grows. That is because there then is no time for any recasting and reinterpretation. Further, as the quantity of data grows then no one can afford the cost and resources that would be required to take the millions and millions of data assets and translate them according to everybody's needs.

The Air Force study and conclusion that they were spending $175 million per year was well before the Internet really started to flourish. Further, it was well before the volume of data started to exponentially grow. So too would grow the vast quantity of data assets. In short, IESSs are essential to manage data.

IESSs that include EIDs enable the data within the IESS to be fully integrated with all other data and of course matched against what is thought to be authoritative data sources. Any differences need to be resolved and updated. Finally, all this data movement, bandwidth permitting, would be through XML wrapped data streams. In short, one more time, these four go hand and glove and together make Net-Centricity possible.

4.0 Enterprise Identifiers (EID)

An enterprise identifier is a guaranteed uniquely identifying value for an asset across an enterprise. The use of Enterprise IDs will ensure exact record-matching among heterogeneous databases even when the databases were designed independently.

EIDs are not synonyms for primary keys nor are they appropriate for all circumstances. Further, not all tables associated with an asset may require an EID. There are two fundamental cases that frame the proper use of EIDs. First is the case where the asset is the dominant purpose of the database, and the second case is when the EID is merely a reference to an asset that has been employed in the database.

In regards to the first fundamental case, dominant purpose. Suppose there is an HR database. It's dominate purpose is the human capital management. The EID seed would be the for Person, and the incrementor would be for the specific individual. In this situation, there are two subcases. First, the database is brand new and is being created from scratch. In this case, the EID could well be the primary mechanism for row identification. Thus, "the" primary key column for an individual would be Person_EID, and all relationships based on that table would have foreign keys such as Skill_Assessment_Person_EID_Reference where it's column is Person_EID.

In the second subcase, there may already be a very elaborate set of internally generated keys within an existing database. In this case, the EID for a specific individual would only have to reside "alongside" the naturally existing HR EID, which might be Employee Number or Social Security Number. The EID would not necessarily be required to become the mechanism for internal relationship mapping of all rows across all tables.

The second fundamental case is that the asset is merely referenced. Thus, the asset's primary database is clearly "elsewhere." In this case, referencing the asset from the "foreign database" would be best done through an EID. This is because the value and meaning of that key is clearly known to the overall database environment, and given that the EID's data for both the seed and the incrementor has been established and is programmatically available, then the mechanism for access should be well within available technology.

4.1 Rationale

Over the years, commonly known assets within the U.S. Department of the Defense have been identified differently. For example, is the unique identifying "number" of an individual the person's name (first & middle & last) along with birthdate, etc., or is it an Employee Id number within the context of the enterprise (a serialized number), or is it the person's Social Security Number. Even within a most common "number," Social Security Number, is it represented as three distinct numbers separated by dashes, a single 11 character string, or 9 distinct numbers. Further complicating the Social Security Number issue is that these numbers are now being reissued.

The same situation exists with almost every different type of enterprise asset, for example, organizations, facilities, physical assets, abstract assets, events, and the like. Making that situation even worse is that different systems and databases across different projects may have made the identifier different. Worse even still is that they have made the identifier the same but have assigned different assets to the same identifying number.

Unless the “identifier problem” is fixed the following problems will continue:

- Assets will be identified differently within different systems thus preventing comprehensive reports about those assets.

- Different assets will be identified using the same identifier thus causing asset reporting to be clearly wrong.

- Requirements for building comprehensive collections of assets that are each known under different identifiers will range from difficult to impossible.

A major challenge facing the enterprise is achieving understanding-based data interoperability. Without it, the semantic web is valueless. To effectively perform many of their critical tasks, knowledge workers need to accurately and rapidly interpret the data they receive across system or organizational boundaries.

At least two capabilities must exist for the knowledge worker to efficiently and reliably interpret the data received from others. First, critical Logistics and C2 (Command and Control) data must be integrated (i.e., must send/receive both "status" and "activity" information). Not only does the person require a unit's status, but also needs to know the impact of logistics on future operational readiness. Logistics information exchanges must be fully integrated into the C2 network down to the tactical level. What is needed is a capability that provides logistics information to the knowledge workers, while simultaneously fulfilling all logistical functions in support of the their needs. Second, the capability must exist to quickly restructure organizations to respond to rapidly changing situations. This requires a capability of accessing the three principal force package data objects (the organization structure, personnel, and materiel) with confidence in their accuracy and reliability.

In general, enterprises are often unable to establish effective and enforceable data standards and processes for interoperability. Because of the large number of information systems containing inconsistent data structures and meanings, the current situation is such that critical data is not available in an accurate and timely manner. Untimely delivery of often inconsistent data in poorly designed information exchanges is largely the result of a lack of a single, universal approach for information identification management.

In regards to inaccurate and untimely information, the knowledge workers need information about the both local and adjacent envioronments to then have situational awareness. This requires that each resource be uniquely identified so that information about them can be queried

and displayed from within local or collaborative information exchanges. Without crosss-functioinal common identifiers, systems might have to store and update half a dozen or more different homegrown tables to identify units. This would be analogous to each U.S. State maintaining its own Zip code directory, which is manually changed/updated every time the force structure changes. Such a situation would be neither efficient or effective.

Many identifiers were standardized under DoD's Data Administration Program and in accordance with documents related to its DoD directive 8320.1. In fact, there are 1,938 approved standard data elements named Identifier in the DDDS. Although many of these have "real world" domains and are implemented in physical databases, many were merely a number that was only conceptual and lack any implementation plan (e.g., Plan Identifier (#5644), Task Identifier (#9282), Person Identifier (#11185), Document Identifier (#9643), Action Identifier (#9904), Situation Identifier (#9909), Action-objective Identifier (#11173), Facility Identifier (#11179), Feature Identifier (#11180), Materiel Identifier (#11182), Capablity Identifier (#11287), and so on).

Looking at the above examples, we can see that these identifiers do not represent trivial objects of interest. Rather they are all intended to provide identifying information critical to uniqueness and interoperability. Many of these identifiers are auto-incrementing numbers without any degree of persistence outside the database environment. In these cases it is perfectly acceptable, for a technical point of view to have the same identifier value related to different objects because persistence and uniqueness was never to extend outside the local database environment. While such schemes are perfectly acceptable in stove-pipe environments, they are completely unacceptable in understanding-based data-interoperability environments. COI environments are both horizontally and vertically integrated across the enterprise.

In a database they are designed to identify a unique occurrence of a plan, task, person, etc. However, when several roles relate to the same object and information about similar objects is shared across functional areas, overlapping meanings can cause problems in identifying object uniqueness. For example, Social Security Number (SSN), is a unique way to identify a U.S. citizen (PERSON), but then so is patient number in a closed hospital environment.

Many of these identifiers (e.g., Organization Identifier, Unit Identifier) exist in the DDDS with no domain they can call their own, because they co-exist with other identifiers whose domains are redundant or partially overlap with theirs. There is no one standard business-based domain for each of them. For example, there are many methods for identifying organizations and units. However, there is no single representation of the concept of organization and no universally accepted unique identifier for it.

The lack of a single, universal approach for information identification management has resulted in too much of the wrong type of data vying for the limited bandwidth available on the battlefield. Clearly, if the interoperability goals are to be achieved, universal, unique identifiers with a common format must be established.

4.2 Technical Construct

The general proposal for an enterprise identifier is that it is first and foremost an immutable numeric value. Further, that this number consist of two 32 bit integers. The first is called the EID seed. The second is a numeric assigned incrementing number that has no intrinsic meaning or value outside of identification. There are no meaningful operations permitted on EIDs such as intervals, sums, or other relational operations such as Greater Than, Less Than, etc. While such operations cannot be prevented, no meaning of any kind can be assigned to or inferred from any result. In short, the EID is a pure numeric identifier that is assigned to a semantic concept. Nothing more and nothing less.

Prototype EID Seed Servers (ESS) have been successfully built that demonstrate an approach based on using 32-bit seed numbers, to serve as prefixes which, when concatenated with another 32-bit suffix, produces a valid EID. Adoption of EIDs is not only well within the technical capabilities of modern information systems, but also can be accomplished with minimal fiscal and performance impact to any system based on available commercial Database Management System (DBMS) technology.

In addition to just EID value issuance and then management, the EID concept should expand to contain an appropriate quantity of “real data” about each “seed,” and then “real data” about each “increment.” This data would clearly be both ADS type data and also IESS reference data.

Although adoption of EIDs has the potential to provide significant cost savings, there are a number of risks associated with its implementation that need to be recognized and mitigated. Specifically, implementations that do not adhere to an enterprise EID strategy will degrade, rather than enhance, interoperability. In order for EIDs to facilitate data exchange and improve data integration and interoperability, an enterprise-wide approach to coherent development, implementation, management, and governance must be employed.

For legacy systems that absolutely cannot change to the EID concept, an IESS can then be built that can be “bolted” onto the legacy application that would contain both the legacy identifier value matched to the EID identifier. While this would lessen the performance of the legacy system by some small amount, it would nonetheless enable that legacy system to employ EIDs in an indirect manner.

4.3 Value to Net Centricity

Table 5 provides the value to Net Centricity from Enterprise Identifiers.

Enterprise Identifiers	
Net Centric Data Goal	**Value description**
Make data visible	Enterprise identifiers enable data asset data to be clearly visible because search engines can be built to "seek out" the data assets regardless of their local identities.
Make data accessible	Enterprise identifiers enable assets to be found and accessed by merely knowing the enterprise identifier. Additionally, all instances of a enterprise identifier seed can be discovered.
Institutionalize data management	Enterprise identifiers harmonize the unique identifiers for classes of assets and also for all identifiers across multiple classes. This more easily allows collections of classes of assets to be brought together for comprehensive reporting. If EIDs are employed to better identify metadata then when an XML schema and/or XML wrapped data employs EIDs the true meaning of the data can be more easily known through quality engineered metadata repositories.
Enable data to be understood	Enterprise identifiers enable data about assets to be quickly located and brought together. Given that the data supporting the EID seeds and data supporting the incrementors both precisely define the data asset classes and then the data assets themselves, then when data brought together from the various information systems
Enable data to be trusted	Enterprise identifiers enable data to be trusted because there will then be a mechanism that can be counted upon to be able to discover all data related to an asset. That way a complete report can be created so that a comprehensive judgement can be determined.
Support data interoperability	Enterprise identifiers greatly enable interoperability because the database and system specific formats and semantics do not have to be remembered and employed to accessed assets.
Be responsive to user needs	Enterprise identifiers and their supporting data along with all other data in various databases and systems enable users to satisfy their data needs. Enterprise identifiers can be employed across communities of interest. This enables users to cross all the databases and systems.

Enterprise Identifiers	
Net Centric Data Goal	**Value description**

Table 5. Value to Net Centricity from Enterprise Identifiers.

4.4 Enterprise Identifier Implementation Process

Accomplishing EIDs requires these high level processes:

- Engineer EIDs and Support Data
- Create EID Enviroment
- Create EID Creation and Assignment Environment
- Identify Assets Requiring EIDs
- Assign EIDs
- Maintain EID Environment

4.4.1 Engineer EIDs and Support Data

Enterprise identifiers have three sets of data. That which is associated with the seed, that which is associated with the incrementor, and transactions associated with the data asset for which the EID is a surrogate. The first two sets of data are mainly about describing the asset directly associated with the EID.

For example, suppose the EID was for a M1A2 Abrams battle tank. The data associated with the seed could well be all design, construction, armor, and weapons, and maintenance data associated with every instance of an M1A2 Abrams battle tank. This data is both an IESS and also because of its definitive status, an ADS.

Similarly, there could be a similar set of data associated with that specific instance of the M1A2 Abrams battle tank. Included for example, might be its manufacturing, configuration, and initial testing. The bright-line boundary of this initial data might be all the data associated with the M1A2 Abrams prior to its delivery to the Army. This data too would both be an IESS and also because of its definitive status, an ADS.

There after, there may be data associated with all transactions associated with the life cycle of that tank. For example, manufacturing, testing, deployment, maintenance, modifications, battle damage, and ultimate disposition.

The data associated with the transactions associated with the data asset could well be descriptive rows of data and then some form of link to obtaining this information. This data could also

extend to mechanisms for specifically extracting this data. For example, there could then be IESS portals that would respond to pre-engineered services that respond with streams of data as the response.

Each class of transaction data could also have an IESS associated with it in any of the three forms described above. Further, there might be systems information associated with the EID so that data access agents would know "where to go" to obtain certain classes of data.

In large measure, the data associated with the EID is a set of IESSs, and the data contained in these IESS databases would certainly be authoritative. How the data would be extracted and shipped around may be through XML.

4.4.2 Create EID Environment

The EID environment is itself one or more database applications that consist of mechanisms to create EID seeds and then collect the data associated with the asset associated with the seed. Thereafter, there must be a mechanism to issue the EID incrementors to authorized persons/organization allowed to assign full EIDs. That EID creator must then collect and enter the data associated with the specific asset instance. Finally, there needs to be a mechanism to collect both the description of data asset data and the mechanisms that can be used to access that data.

4.4.3 Create EID Creation and Assignment Environment

The EID creation and assignment environment consists of the procedures and processes whereby an organization determines the need for a particular EID within a specific EID class (that is, seed). The software environment thus needs to have pre-loaded EID seeds so that the person needing the EID can first select the seed, and then request the incrementor.

4.4.4 Identify Assets Requiring EIDs

The assets that are going to be tracked first have to be identified. The organization that is the authoritative data source for that asset then has to determine the appropriate data and enter it into the EID database. The objective here is to have a database table, an Asset Id Table, that contains sufficient information that would enable EID assigners the ability to recognized that a specific asset instance should be then assigned a particular EID. Such information may include its description, manufacturing date, or other identifying information.

4.4.5 Assign EIDs

The most difficult issue associated with EIDs is finding the data associated with the data asset and then assigning the EID to that data. There are generally two methods of assignment. The first is to modify the database table that contains the data asset transaction data, say, its location history, and add an EID column. At that point, an update program would have to be created that would allow some sort of matching between the Asset ID table and the asset's transaction data. A second strategy would be to create an miniature IESS that would contain only those columns necessary to match an asset's already known identifier with the EID.

4.4.6 Maintain EID Environment

Maintaining the EID environment generally consists of tasks that are normal for any database project. Components of the maintenance might include adding XML schemas and services that would facilitate access to the asset transaction data, or adding or modifying database tables that would provide additional descriptive material about the EID seed's asset class or the EID incrementor's specific asset.

4.4.7 EID Implementation Summary

Enterprise identifiers have been necessary ever since there has been a need for data sharing. EIDs have existed for a very long time, just under different names and formats. The data asset products that need to be constructed in support of enterprise identifiers are listed in Table 6.

Data Asset Products for Enterprise Identifiers		
Prod Ref	**Architecture View**	**Architecture Product**
AV-1	Overview and Summary Information	Mission, **Database Domains**
AV-2	Integrated Dictionary	Business Terms, **Data Element Model**
OV-1	High-level Operational Concept Graphic	Mission, **Database Domains**, Functions, and Organizations
OV-5	Operational Activity Model	Business Function, Business Events, **Physical Data Model, and View Data Model**
OV-6c	Operational Event-Trace Description	Mappings Regarding Business Functions, Business Events, **Physical Data Model**, **Views, Data Integrity Rules**, Business Information Systems, and **Database Objects**.
OV-7	Logical Data Model	**Data Element Model, Conceptual Data Model, and Logical Data Model**.
SV-1	Systems Interface Description	Business Information Systems, **Views, and Inter-view Mappings**
SV-3	Systems-Systems Matrix	Business Information Systems, **View Models, and Physical Data Models**
SV-4	Systems Functionality Description	Business Information Systems, **Database Objects Model, and Logical Data Model**
SV-5	Operational Activity to Systems Function	

Data Asset Products for Enterprise Identifiers		
Prod Ref	**Architecture View**	**Architecture Product**
	Traceability Matrix	
SV-6	Systems Data Exchange Matrix	Business Information Systems, **Views, and Inter-view Mappings**
SV –10a	Systems Rules Model	Business Information System, **Database Objects, Data Integrity Rules and Logical Data Model**.
SV-10b	Systems State Transition Description	Business Events and **Database Objects**
SV-11	Physical Schema	**Physical Data Model and View Data Model**

Table 6. Identification of Data Asset Products for Enterprise Identifiers

Very common examples are the Social Security Number (SSN) and the Vehicle Identification Number (VIN). Even though both these "numbers" specially formatted, not really numeric, and are information bearing, they are EIDs nonetheless. Each is captured and employed in various IT systems to uniquely identify data associated with a person or an automotive vehicle.

The EID in a data management program is generally formatted as two 32 bit numbers that carry no information intrinsic to its numeric value whatsoever. Thus, the use of EIDs across the systems that employ it always know it's data type (integer), its name (EID), and that it's a unique surrogate for the asset it represents. Because there's no inherent name, e.g., SSN or VIN, the EID may become part of the field's name, as in Organization EID, Location EID, or Equipment EID. To "learn" more about that EID class, a query would have to be launched to the EID server and information about the EID's seed would be forthcoming. For example, it might come back with name and descriptive information about the organization's class. For example, that might be about an organization. A subsequent query for the full EID might then give information about the specific organization that exists within another organization. Thereafter, a search launched for all information about the initial organization might give its location, construction, and other transactional information about the organization from the time was created to the present time.

EIDs should be assigned to which ever assets are to be tracked across multiple AISs. The classes of assets that are commonly associated EIDs are physical assets like the M1A2 Abrams battle tank described in earlier.

EIDs may also be assigned to abstract assets like organizations and financial accounts. EIDs may also be associated with IT assets like ADS data instances, and physical data model DBMS table columns. When associated with ADS data instances, then the precise nature of any reference data stored in a data asset can quickly and definitively known.

EIDs employed as identifiers of physical data model DBMS table columns offer a more abstract but very valuable purpose. For example, it is proposed in the ANCDM that every XML element have an attribute that contains the EID of the physical data model DBMS table column from which it was generated. This would enable users of an XML schema to definitively know which physical data model, which DBMS, which database table, and which database table column was the source for the data value. Once this is know, queries could return information associated with

the physical data model's parent logical data model, parent conceptual data model, parent ISO 11179 data element, and associated value domains including values and meanings. Returned as well could be all definitions and other descriptive information including what missions, functions, and organizations are involved in the XML schema. If the EID concept were extended to AISs then all the information about the AIS that was involved in constructing the XML schema could be returned including all formulas, dates and times of execution, and the like. In short, the EID concept, if properly engineered and deployed is very powerful.

4.5 Summary

Enterprise Identifiers are essential to uniquely identify assets. Rarely are data records so complete that the class of asset is fully known or that a specific asset is identified. Thus, when data includes enterprise identifiers, both the data surrounding the asset class and the asset instance are then automatically capable to be known. It is therefore important to have a significant quantity of descriptive and characteristic data supporting each EID class. Similarly, it's important to have the necessary and sufficient data supporting the specific asset instance. When this is done, then all that class and instance data will not have to accompany the myriad of location, transit, cost, valuation, logistics, acquisition, maintenance, modification, update, and disposition data that ultimately become the asset's life cycle of transformations expressed in date and time based data transaction format.

EIDs, when coupled with the location of all shared asset data within IESSs, and the ADS data that provides for definitive states about the assets, greatly assist in the accomplishment of the comprehensive understanding, analysis, and decision making. It is necessary that the transaction data, wrapped in XML contain the EIDs, and also that there also be metadata based EIDs so that the complete context of the database and the information systems that contained the asset transactions were also known. Again, these four go hand and glove and together make Net-Centricity possible.

With a quality engineered EID concept, the six interrogatives can be answered about any critical asset: What, where, when , why, who, and how.

5.0 Extensible Markup Language (XML)

XML is a text-based format that is being used for structured document and data exchanges. XML is used in this way to describe and couch data in support of non-proprietary data exchanges that are independent of any DBMS and/or database application. The XML data streams can also be displayed through Internet browsers. XML tags need to be based on standardized data for maximum reuse.

5.1 Rationale

Almost in parallel with the evolution of data processing has been the trek to remove unwanted bindings between applications and data. In short, a trek towards data independence. During the 1960s, data was completely defined within the context of the application program. This caused serious problems to applications whenever any insignificant data design change happened. When changes happened, it was not uncommon to have to recompile hundreds of computer programs. In short, there was a very tight binding between data and the program.

For this very reason, certain classes of database management systems (DBMS) were invented. The definition of data was the province of the DBMS. Then, the application system accessed the database through the DBMS. This insulated the application from minor changes in the database's design. This started increasing the level of data independence.

Still, the application had to know about the structure of the database because there had to be detailed programming language commands in the application to properly traverse the database. This was regardless of the data model of the database. Relationship navigation binding was as much a problem with relational DBMSs as it was with other data model DBMS (network and hierarchical). In short there still wasn't complete data independence. In response the DBMS community invented the View. It is a DBMS object that enables the database selection and navigation logic to be removed from the application program and installed in the database's schema. This increased the level of data independence because the application program was no longer bound to the database.

But there remained some level of binding. It was the binding of the data's format with the application program. The data still had to be formatted such that the application program knew what the values meant both in terms of data type and precision and also in terms of position within the data stream.

So, a next step in the drive towards data independence was to make the data available from the database and the application program in a completely independent form. Hence, XML. The application programs retrieve data, form it into XML streams, and then posts the data to some space. Thereafter any other program can access the XML stream of data, "understand" it because of the XML tags, and then employ the data within the application program. Has complete data independence been reached? No.

Still, there was some remaining binding. It was the binding that the application program still had to "know" where the DBMS was so that data needed by an application program could be obtained without knowing the "location" of the DBMS. This currently is done on personal computers by "telling" the database application the "location" of the DBMS connection process (known as ODBC). The solution to this final level of binding is called Service Oriented Architecture. The goal of this strategy is for the application program able to issue a request for data (or to store data), and the SOA environment be able to locate the DBMS and obtain (or store) the data. When the SOA environment is accomplished then there will finally be a high level of data independence.

However, despite achievement of all these levels of data independence, if there is not a comprehensive set of data semantics, granularity, and precision on the part of an accessing application, the result will still be a "blank stare." In short, without consensus based semantics, granularity, and precision, there will only be connectivity-interoperability. There will not be any level of understandability-interoperability along with minimum complexity and latency.

Although XML has the potential to provide significant cost savings, there are a number of risks associated with its implementation that need to be recognized and mitigated. Specifically, implementations that do not adhere to an enterprise strategy will degrade, rather than enhance, interoperability. In order for XML to facilitate data exchange and improve interoperability, an enterprise-wide approach to standard XML development, implementation, namespace management, and governance must be employed. The following questions need to be answered:

- Does XML by itself constitute the silver bullet for achieving understanding-based data interoperability? Can XML solve our net-centric issues for data without standardization?

- Or must we also have a multi-component strategy consisting of (1) enterprise identifiers (EID), (2) XML, (3) information exchange standards specifications (IESS), and (4) authoritative data sources (ADS), to exchange data effectively?

Guidance from the GAO clearly addresses these questions. The United States' Government Accounting Office, in the document *Report to the Chairman, Committee on Governmental Affairs, U.S. Senate, Electronic Government: Challenges to Effective Adoption of the Extensible Markup Language, April 2002, GAO*, stated in Chapter 4, Conclusions and Recommendations, the following:

> XML's greatest benefits accrue when organizations, such as government agencies, use standard data exchange procedures and agree on standard data definitions and structures. Effectively using XML as a means to share data among disparate systems across the federal government will require agencies to conform to a range of technical and business standards. While XML's technical standards are largely in place, important business standards – including many planned standard vocabularies – have not yet been completed, and in some cases, standards development to date has resulted in incompatibilities. To the extent that these business standards address government needs as they are developed, government agencies will likely have less of a need to develop their own nonstandard data vocabularies and structures (p. 58).

> Given that a complete set of XML-related standards is not yet available, system developers must be wary of several pitfalls associated with implementing XML that could limit its potential to facilitate broad information exchange or adversely affect interoperability, including (1) the risk that redundant data definitions, vocabularies, and structures will proliferate, (2) the potential for proprietary extensions to be built that would defeat XML's goal of broad interoperability, and (3) the need to maintain adequate security (p. 58).
>
> XML's larger promise of facilitating data exchange across broad domains (such as an entire agency, a group of agencies, or a set of external stakeholders and client organizations) will be difficult to realize until critical data elements and structures are identified and standardized across entire agencies and communities of interest. This task of identifying and standardizing critical data elements and structures is part of an agency's larger task of developing an enterprise architecture. Well-planned enterprise architectures can also promote the adoption of flexible implementations that can be modified in the future to conform to commercial standards that become established over time. Thus, agency enterprise architectures are key building blocks to effective government wide adoption of XML (p. 59).

XML (eXtensible Markup Language) employs tags to represent both the data itself and how it appears on a Web page. It can be used to describe the contents of both structured and unstructured data. It is an inexpensive mechanism for data exchange. Some call it the lengua franca of the Web. Given these and other capabilities of XML, the following questions arise when looking for a means to achieve understanding-based data interoperability:

- Does XML by itself constitute the silver bullet for achieving understanding-based data interoperability?
- Can XML solve our net-centric issues for data without standardization?

Already some communities such as the GAO, the DOD architecture community, and other communities are learning that XML is no silver bullet. Enterprises have realized that XML must reside within a multi-component approach that relates XML to three other components: Information Exchange Standards Specifications (IESS), Authoritative Data Sources (ADS), and Enterprise Identifiers (EID). Specifically, XML must conform to IESSs for maximum reuse and understanding-based data interoperability impact. XML can serve as an enabler for ADSs, through the definition, exchange and maintenance of coded domains. With advances in compression techniques, all database-to-database exchanges via XML in conjunction with EIDs (Key management) to maintain referential integrity among all participating systems may become practical.

Mr. Craig S. Mullins, in Database Administration: The Complete Guide to Practices and Procedures, Boston, Addison-Wesley, 2002, Pages 551 and 553, states:

> In short, XML allows designers to create their own customized tags, thereby enabling the definition, transmission, validation, and interpretation of data between applications and between organizations…XML is quickly becoming the de facto standard for application interfaces…
>
> The capabilities of XML will cause it to gain acceptance, regardless of any drawbacks. And XML does have drawbacks, including:

- An increase in the size of data files encoded as XML documents due to the metadata XML tags. Larger documents take longer to transfer across the network than smaller files.
- The need for yet another "model" of storing data. If data is stored in relational databases and incorporated into XML documents for sharing with others, you may be able to mitigate this problem.
- The over-hype of XML. There is a lot of confusion surrounding XML in the industry. Some pundits have claimed that XML will provide metadata where none currently exists, or that XML will replace SQL as a data access method for relational data. Neither of these assertions is true.

Mr. Mullins statements are clearly supported by other leaders in the data management industry. See for example http://www.tdan.com/i021hy02.htm. This article's clear conclusion is that XML will have significant value to the enterprise and will increase understandability-based interoperability if and only if it proceeds from a foundation of well engineered data standards such as those described in this paper.

XML, therefore is one component among a comprehensive multi-component understanding-based data interoperability strategy. It also must be understood that the attention paid to understanding-based data interoperability is due to the enormous impact both in terms of cost as well as in terms of decreased efficiency that arises when systems cannot exchange data without the need for system-to-system translators.

In support of this multi-pronged approach, the DoD Architecture Framework states:

> The heart of interoperability is the preservation of meaning and relationships during data exchange (and data reuse). A data model is a structured representation of the data elements pertinent to an architecture, often including the relationships of data. Agreement on a data model is essential to exchange and reuse architecture data, as well as the implementation of architecture databases, regardless of the technology chosen (e.g., relational, object oriented) for building and managing architecture databases. In addition, a common data model can serve as the basis for defining common Extensible Markup Language (XML) tags for data subject to import, export, product extraction, and direct exchange. (DoD Architecture Framework, Version 1.0 (Final Draft), Vol. I, page 5-2)

In short, if XML were to become the sole basis for attempting to achieve understanding-based data interoperability in DoD, the result would not only be more fragmented databases than ever, but also a XML tag standardization trek that is significantly more costly and time consuming than the failed and halted "8320.1" effort. The opportunity to achieve understanding-based data interoperability will diminish. In contrast, if XML is implemented in conjunction with EISS, ADS, and EID, the opportunity to achieve understanding-based data interoperability and net-centricity enterprise-wide will greatly increase.

5.2 Technical Construct

XML should be considered and used as a technology by all programs that exchange information or data to include (but not limited to): (1) messaging; (2) database transfer; (3) office products file transfer; and (4) other system to system data exchanges and processing.

XML need not be used in all cases, but should be considered as a document/data definition format in the same way that HTML is considered and utilized as a display format. Note that the focus of HTML has traditionally been on display (what does the data look like on the screen?), whereas that of XML has focused on content (what is the data?). Yet, common browsers and office applications recognize both formats and the definition of XHTML (XML HTML) makes HTML a proper subset of XML.

When XML is used, there should be discussion among the developers and users of the particular applications community of interest regarding utilization of common tags (named elements) and DTDs (as well as schemas, style sheets, and other XML constructs) where appropriate. Lack of commonality will necessitate the use of translators. XML schemas, of course, must be based on IESS data models.

Newer browsers such as Internet Explorer (5.0 or above) and Netscape Communicator (4.6 and above) work best with XML. XML parsers and editors are available from several vendors (Sun, Microsoft, IBM). Programs may thus need to implement later versions of browsers and appropriate parsers.

In terms of benefits, XML: (1) allows easier transformation and viewing of diverse data; (2) can be used to allow systems to exchange more types of messages; (3) allows easier updating and synchronization of diverse databases; and (4) can be used to mine metadata and request files.

XML has the potential for wide use commercially, within the government, and in military systems. Major software companies—such as Microsoft, Sun, IBM, Sybase, Informix, and Oracle—are integrating XML within their products in the same way that HTML was integrated several years ago. This is because of the fact that XML is a powerful method of structuring data in a text or ASCII file. XML files can handle data/text information as well as pointers to binary files. As such, XML can be important for information exchange between applications and/or between clients and servers.

XML is the unbinding of the data transport layer into ASCII streams of hierarchically organized data tags and values. Heading every XML data stream is an XML Schema that defines the data value stream that follows.

To create a XML stream, the sending system must extract the data from its proprietary format and conform it to the XML schema. While it is valuable to have every data export, save those requiring severe compactness or having to fit with a very restricted bandwidth, conformed to a XML format, there is little to the savings if system to system data interfaces are replaced with

XML'ized system to system data interfaces. In fact, an argument could be made that it is more costly to create such a XML system to system interface because the data first has to be composed into the XML format, and once received it has to be shredded into a database (most commonly relational) format. Finally, it is of little increased value to publish XML schemas as it is to either publish COBOL FDs, or to publish SQL schema DDL.

The value then of XML is when it becomes the data transport layer for IESSs. Because of IESSs, the quantity of composition and shredding operations severely reduces. The XML is icing on the cake. The cake is made from flour. Flour production takes significant wheat cultivation, planting, tilling, harvesting, and grinding. So too does data standardization, which, when accomplished becomes the flour used to bake the case that is iced with "XML."

An additional value of XML is in the XSLT. That is, the XML Stylesheet Language Transformations is now a general-purpose translation tool. The XSLT is supported within the environment through the mapping of a SQL view from one business information system (the sending system) to the business information system of the IESS (the receiving system). Once the view columns are mapped then the XSLT can be generated. This can then support the generalized composing and shredding of XML transmissions from one system to the other.

The value of XML is founded upon the quality of the underlying data models upon which it must be based. When the XML also includes the EIDs of the supporting metadata and/or ADS then receiving systems can query the source metadata and can learn about the authoritativeness of the data source.

5.3 Value to Net Centricity

Table 7 provides the value to Net Centricity from eXtensible Markup Language (XLM).

eXtensible Markup Language (XML)	
Net Centric Data Goal	**Value description**
Make data visible	XML enables assets to be seen from their data assets via data display mechanisms
Make data accessible	XML does not make data more accessible, but once retrieved it makes it easier to display. Additionally, if XML is employed to wrap data exchanges then if the XML is properly supported by consensus based data standards and support importing programs then the data will be more easily be able to be employed.
Institutionalize data management	XML will not improve institutionalization except that data exchange formatting will be easier to construct..

eXtensible Markup Language (XML)	
Net Centric Data Goal	**Value description**
Enable data to be understood	XML will improve understandability if the XML is supplemented by EIDs that point to either seed-based data or incrementor-based data. In that case, there will be further information about the managed asset, and there will be a access mechanism that will enable traversal from data asset about the data to another.
Enable data to be trusted	XML will not directly impact trustability. However if the other three components (ADS, IESS, and EIDs) are all present the retrieved data will be more easily trusted.
Support data interoperability	XML only affects interoperability in that there is a universal format for data transport and display. If the data is not quality data, or based on consensus based semantics, or able to be quickly interrelated through EIDs, then while the data is available it may be conflicting with other data, or not harmonized with respect to semantics, granularity or precision.
Be responsive to user needs	XML only satisfies user needs in that data will be more easily able to be retrieved and displayed. If it is not quality data or based on consensus based semantics, etc., then the data will not be responsive to user needs regardless of its quick access or quantity.

Table 7. Value to Net Centricity from XML.

5.4 XML Implementation Process

Accomplishing XML requires these high level processes:

- Create XML Schemas standards
- Create XML Schemas within IESSs
- Create XML Schema generation process
- Create automatic tagging of data assets
- Create data asset access strategies

5.4.1 Create XML Schemas Standards

Standards for XML schema generation are necessary to then minimize the analysis, either manual or automated that must or can be performed against a library of XML schemas. The XML schema, as the XML data structure representation of the data from the data asset must be sufficiently set out that the type, kind, description, and definition of the data can be sufficiently understood so that it can be selected among many other XML schemas.

When there is insufficient information to fully understand an XML schema, then there will be a link back (possibly through an EID approach described above) back to the metadata environment from which the XML schema was generated. It is expected that such a link back would either provide information about a XML element or attribute, or a XML segment, or a full XML schema.

Included in the standards will be sufficient configuration management information to then indicate, for example, whether the XML schema and associated data is authoritative, date and time of collection, the level of granularity and precision.

5.4.2 Create XML Schemas within IESSs

During the process of creating IESSs, the user community of data providers and consumers will identify their interface requirements and create a full set of XML schemas representing those exchanges. These XML schemas will be cataloged and set into a library. As IESS community users employ data from the IESSs based on these XML schemas they should first register their use so that any possible changes to the XML schema of the IESS can be properly configuration managed.

5.4.3 Create XML Schema Generation Process

The XML schema generation process will be based on configuring the XML schema just as if it were the consequence of creating a SQL View, that is, it will be semantically correct. The data represented by the XML schema and populated into the XML stream will be extracted from the database only through selects and nested selects based on database schema defined relationships.

The process of XML schema generation will be largely automated by picking the database tables and or a subset of columns from the tables that are to participate in the XML schema. Once chosen, then the XML schema will be generated. The names employed in the XML schema will be automatically created according to the appropriate set of rules for composing a XML element or attribute name from a database table column.

If the full data management infrastructure exists upon which XML schemas are then automatically generated, the process of transforming on such XML schema's data stream to

another can largely be automated as well. An XSLT, that is a XML Stylesheet Language Transformation is the transformation of one XML data stream to another via a transformation process. If both the input XML data stream and the output XML data stream are based on XML schemas that are drawn from the data management program infrastructure then if the two XML schemas are logically related via common upper levels of logical, conceptual, or ISO 11179 data element layers then a transformation can be posited. The automatic generation of the XSLT will fail, of course, any of the following reasons:

- The two XML schemas cannot be determined to be logically related, or
- The value domains for the XML data elements are not able to be mapped, or
- The data types are unacceptably different

In such a case, the XSLT can be forced to perform the transformation, but a serious question should be asked first: Are these two XML schemas sufficiently semantically related to support a transformation?

5.4.4 Create Automatic Tagging of Data Assets

Every data asset is to be represented by metadata that is to associated with metadata catalog metadata. To ensure some level of conformity in the metadata catalog entries, these metadata, for database data assets should to be automatically generated based on a localized set of metadata assigned to the database's schema, or its tables, or at a higher level of inheritance than the physical schema. That higher level would come from the logical schema or the conceptual schema level.

5.4.5 Create Data Asset Access Strategies

The access strategy for actually obtaining the data values represented by the data asset has to be engineered such that access can be accomplished either through a browser interface or programmatically. A programmatic interface involves mapping the fields to an internal schematic and then processing the data records.

5.4.6 XML Implementation Summary

It may seem a surprise that there is so little information associated with XML. Just about two pages for its high level process model. That's because the data management program accomplishes all the foundational data management infrastructure on which the XML is then just one small additional component. The real value of XML is that it enables the creation of technology independent formatting of data exchanges. XML further offers the ability through XSLTs of transforming one XML data stream to another. The data asset products that must be involved during the creation of an XML infrastructure and products is listed in Table 8.

If the enterprise has not accomplished all the necessary prerequisite data management infrastructure work, then XML can be brought to bear to create immediate and ad hoc solutions. If such solutions are allowed to grow and propagate, the only thing that will be accomplished is an uncontrolled growth of point-to-point solutions. Additionally, it has been shown in Section 2.3.5 that a XML solution alone is not cost effective over one that is data centric and largely able to be automated. The cost of point-to-point solutions has also been shown to be prohibitively expensive. Based on an extrapolation of the USAF study done in the middle 1990s, DoD is likely spending about $1 Billion per year on point-to-point interfaces. Section 3.2 1 contrasts the point-to-point versus a data centric approach as a foundation for XML creation and management.

In short, the most efficient and effective way to deploy XML is on top of a high quality data management infrastructure.

Data Asset Products for XML		
Prod Ref	**Architecture Product**	**Architecture Product**
AV-1	Overview and Summary Information	Mission, **Database Domains**
AV-2	Integrated Dictionary	Business Terms, **Data Element Model**
OV-3	Operational Information Exchange Matrix	**IESS via physical data model**, with Business Information Systems, Business Event, Business Calendar, and Business Cycle
OV-7	Logical Data Model	**Data Element Model, Conceptual Data Model, and Logical Data Model**.
SV-1	Systems Interface Description	Business Information Systems, **Views, and Inter-view Mappings**
SV-3	Systems-Systems Matrix	Business Information Systems, **View Models, and Physical Data Models**
SV-4	Systems Functionality Description	Business Information Systems, **Database Objects Model, and Logical Data Model**
SV-6	Systems Data Exchange Matrix	Business Information Systems, **Views, and Inter-view Mappings**
SV-11	Physical Schema	**Physical Data Model and View Data Model**

Table 8. Identification of Data Asset Products for XML

5.5 Summary

XML is hailed as the key mechanism to make Net-Centricity come alive. That is of course true only if the necessary and sufficient foundations that support Net-Centricity are already in place. That is, that ADS, EIDs, and IESSs exist and have been properly configured within sophisticated database environments.

6.0 Data Standards Architecture and Implementation Summary

To have complete understandability-based understanding-based data interoperability with minimum complexity and latency, the following must exist:

- Enterprise Identifiers
- Authoritative Data Sources
- Information Exchange Standards Specification
- XML data environment

The enterprise identifiers enable unambiguous identification of either data or metadata regardless of what these may be named within technologically constrained environments. Authoritative data sources provide the critical data upon which data is characterized, summarized and upon which key decisions are made. Information exchange specification standards provide for the shared "data understandings" among members of a community of interest and between communities of interest. Finally, XML provides for the unbinding of data within their proprietary formats and for the transmission and transformation of that within data-interoperable environments.

Implementing Net-Centric Data Management involves accomplishing the four data standards within the context of either specific projects, or given shared data needs across a collection of projects, within communities of iInterest.

Without these four data standards, and without a development environment within which these standards and their metadata can flourish and grow, Net-Centricity will surely fail. But more importantly, the understandability-based interoperability that is easy to declare, but difficult, time consuming, and laborious to achieve will not also succeed. The cost of not having understanding-based interoperability with minimum complexity and latency ranges from diminished information timeliness and value to fratricide. Failure to achieve cannot be allowed.